Milestone Report

Yuqi Gong, Tianyi Chen

Progress so far:

As mentioned in the proposal, algorithm design is the main focus of this project, so we spent a lot of time digging through literature regarding Delaunay Triangulation. We mainly looked at three categories of sequential algorithm and corresponding attempts at parallelizing them. Here is a brief summary:

(1) Duality of Delaunay Triangulation and Voronoi Diagram.
Voronoi diagram, which partitions the space into regions closet to each of the given points, is the dual graph of Delaunay triangulation. Thus, efforts have been made to compute the Voronoi diagram in parallel, and reconstruct the Delaunay triangulation from that. In particular, we noticed that another group from previous semester utilized the "jump flood" algorithm to compute the discrete Voronoi diagram, and achieved significant speedup on Cuda. So, we decided to take on a different approach.

(2) Incremental Insertion.
The sequential incremental insertion algorithm is based on the invariant of Delaunay Triangulation, which says no circumcircle of any triangle contains another point. So, we just incrementally add in new vertex and replace triangles whose circumcircle contains it. The parallel version seeks to speed up the process by observing that most of the updates only require "local" changes. So, multiple insertions can be performed in the graph at the same time. The actual algorithm is more refined than this, which we'll discuss in the final report.

(3) Divide and conquer.
The vanilla divide and conquer algorithm partition the given point set in half, solve the Delaunay triangulation, and use a "clever" merge trick to achieve O(nlogn) time complexity. Even though divide and conquer algorithm are generally suitable for parallelism, in this case, the merge step is inherently sequential, and has to be performed on only one processor, which greatly limits the speedup. The workaround we found is to use projection to divide the points beforehand, such that the boundary will always be part of the Delaunay Triangulation, thereby avoiding the merge step altogether. If the initial split can be done efficiently, the potential speed up could be very promising.

Preliminary Results:

For this milestone, we tried to implement the parallel version extended from incremental insertion algorithm **from scratch.** Right now, the algorithm can produce valid results on some input (up to 200 vertices). However, there seems to be a minor bug that's causing issue on more dense input points. The next step will be to eliminate this issue and derive some meaningful speedup graphs.

Concerns:

Delaunay Triangulation is a non-trivial problem in computational geometry. As a result, the runtime of any algorithm is highly implementation-dependent. Clever preprocessing or use of particular data structure can influence the runtime. It might not necessarily affect the

asymptotic time complexity, but any constant factor optimization matters when we are talking about speed up from parallelization. This is exactly why we decide to implement the whole thing **from scratch**. We realize that this way the speedup would not be representative if the parallel version is not work efficient. Therefore, we'll make sure to discuss any implication of this decision on the final results. But for the incremental insertion algorithm, this shouldn't pose any serious problem, as the algorithm is shown to be work efficient.

Revised Schedule:

WEEK 3.0 (11/23 - 11/26): Sequential Version Debugging

- Have a working sequential implementation based on incremental-insertion.
- Can generate visual diagram.

WEEK 3.5 (11/27 - 11/29): Implementation of 1st Parallel Version

- Complete parallel implementation based on incremental-insertion.
- Complete correctness debugging.

WEEK 4.0 (11/30 - 12/3): Performance Debugging

- Debug and improve the performance of the first parallel implementation.
- Generate profile diagrams for poster session.

WEEK 4.5 (12/4 - 12/6): Implementation of 2nd Parallel Version

- Understand the divide-and-conquer algorithm.
- Start parallel implementation based on divide-and-conquer.

WEEK 5.0 (12/7 - 12/10): Implementation of 2nd Parallel Version (Cont.)

- Complete parallel implementation based on divide-and-conquer.
- Complete performance debugging.

WEEK 5.5 (12/11 - 12/13): 2nd Performance Debugging

- Debug and improve the performance of the first parallel implementation.
- Generate profile diagrams for poster session.

WEEK 6.0 (12/14 - 12/18): Preparation for Poster Session

- Gather texts and diagrams for the poster.
- Create the poster for the final presentation.

Poster Session:

We plan to show the Delaunay Triangulations generate by our algorithms with different number of input points.

In addition, we will present visualizations of how the algorithm works.

Finally, we will present various performance diagrams, such as speedup diagram, thread profiles, and etc.